For converting the infix string to postfix, two linked list’s are created. One for containing operands and one for containing the operators.

The getType(char token) function is made with a switch case instead of if/else nesting. This is decided due to the readability of the code.   
The types decided to be operands, operators and start/stop parenthesis. The operators are divided into two levels of precedence because the string is only allowed to contain the operators + - \* /.  
+ and – are classified as operatorPrecedence0, and \* / are classified as operatorPrecedence0.

When given an operand, it is always put in the operandstack. For readability it is put at the end of the stack.

When given an operator with lowest precedence, the operator stack is consulted:  
If the operatorStack is empty, the operator is pushed to the stack.  
else, if the operatorStack contains another operator with lowest priority, the operator from the stack is popped and put in the postfixResultStack, the operator at hand is pushed to the operatorStack.   
else, if the operatorStack contains a starting parenthesis, calculations should proceed as usual and the operator is pushed to the operatorstack.  
Else if the operator stack contains an operator with higher precedence, the higher precedence operator is popped to the resultstack and the operator at hand is pushed to the operator stack.

When given an operator with highest precedence, the operator stack is consulted:  
If the operatorStack is empty, the operator is pushed to the stack.  
else, if the operatorStack contains an operator with lower priority, the operator at hand is pushed onto the operatorStack.  
else if, the operatorStack contains a starting or stop parenthesis the operator at hand is pushed to the operatorStack.  
else if the operatorstack contains another operator with highest priority, the operator from the operatorStack is popped to the resultStack and the operator at hand is pushed top the operatorStack.

If the operator at hand is a starting parenthesis it is pushed to the operatorStack.

If the operator at hand is a stop parenthesis the operators in the operatorStack is popped to the resultstack until the first operator in the operatorstack is a starting parenthesis. The starting parenthesis is then removed from the operatorstack.

After the whole expression is read, we empty the OperatorStack to the resultStack.

When calculating the postfix expression an operand stack is used to store the operands which in the end will be the result.  
While the Resultstack is not empty the first character is read and handled in a switch case.  
When an operand is encountered it is popped from the resultstack and pushed to the operandstack.  
When an operator is encountered the operate function is called.  
The operator function takes 3 parameters: The operator and the previous two operands from the operandstack. These two operands are popped from the operandStack. The result is a double that is pushed back into the operandStack.

When the last operator from the postfixResult is read, there will be only one number left in the operandStack which will be the result of the calculation.